Домашна задача 3

Тијана Атанасовска (196014)

Барањe:

Податочното множество (од датотеката diabetes \_ binary \_ health \_ indicators \_ BRFSS2015.csv) потребно е да го поделите случајно во 2 датотеки:

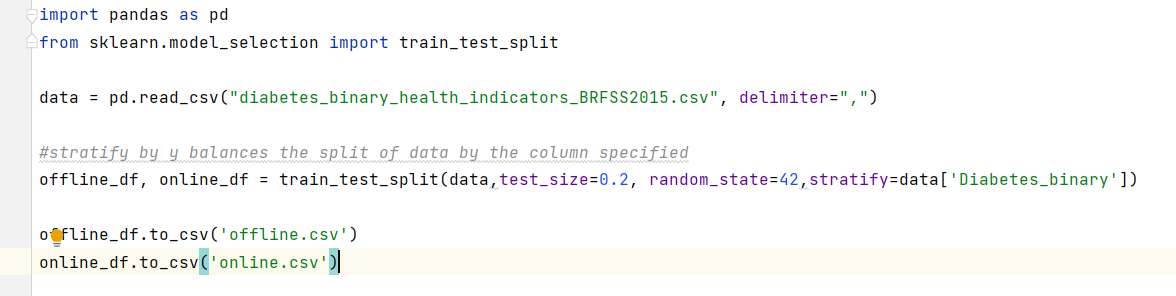
offline.csv (80% од големината на оригиналното множество)

online.csv (20% од големината на оригиналното множество)

Во поделбата внимавајте да се задржи соодносот на класите. Ова поделба може да ја направите на начин по ваш избор (препорачуваме користење на python и методот train\_test\_split од библиотеката sklearn).

Решение:

Се вчитува датасетот со користење на Pandas. Потоа се дели на 80%-20% со балансирање според колоната за класата (користење на stratify).



Барањe:

Offline фаза

Со помош на податоците во датотеката offline.csv потребно е да креирате DataFrame/Dataset во Apache Spark апликација. Истиот DataFrame/Dataset потребно е да го трансформирате со трансформации по ваш избор со цел врз DataFrame-от да може да извршувате алгоритми од машинско учење.

\*\*Трансформациите направете ги во методи кои би можеле после да ги извршите по втор пат врз податоците од датотеката online.csv.

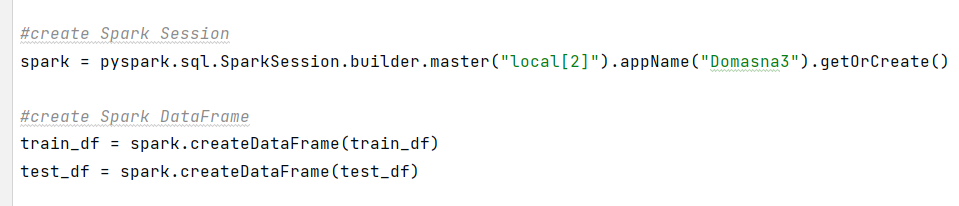
Потребно е да обучите 3 модели за класификација со различни комбинации за хиперпараметрите за истите модели. Методот којшто ќе даде најдобри резултати по метрика F1 потребно е да го зачувате (серијализирате) локално во некој директориум. При избор на најдобриот модел потребно е да користите некоја стратегија за избор на најдобар модел (train/test split, cross-validation, K-fold)

Решение:

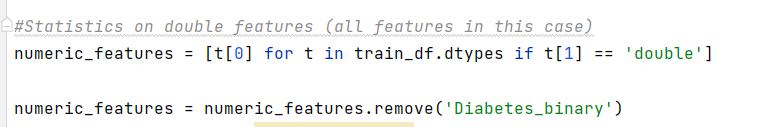
Се вчитува датасетот offline.csv и повторно се дели на 80%-20% според колоната за класата.

Се креира Спарк апликација со име Domasna3, сетирана на 2 cores -> local[2].

Oд вчитаните Пандас датасети се трансформираат во Спарк датафрамес.

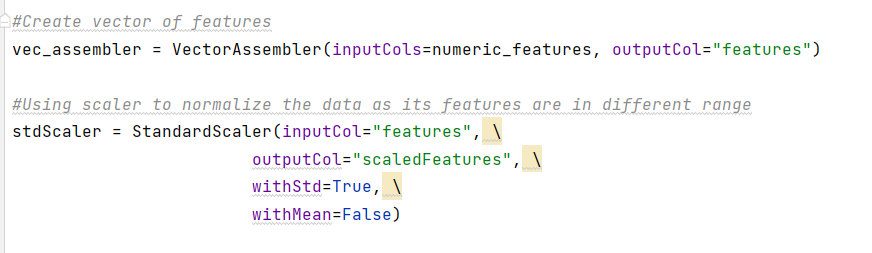


Се земаат само нумеричките колони. Доколку имаше од друг вид, би требало да се индексиират во броеви.



Се креира ВекторАсемблер кој прави една колона од влезните колони и е првиот дел од трансформациите.

Се креира и објект од StandardScaler за скалирање на вредностите, поради различноста на рангот на колоните.



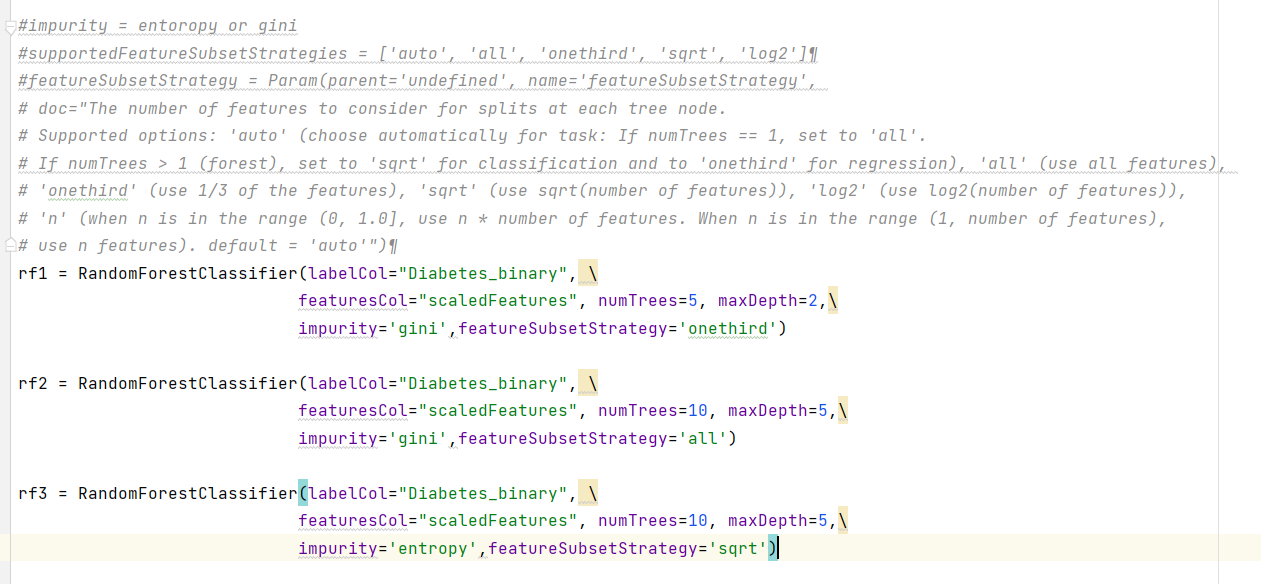
Модели:

Тренирав вкупно 9 модели – 3 модели со различни параметри.

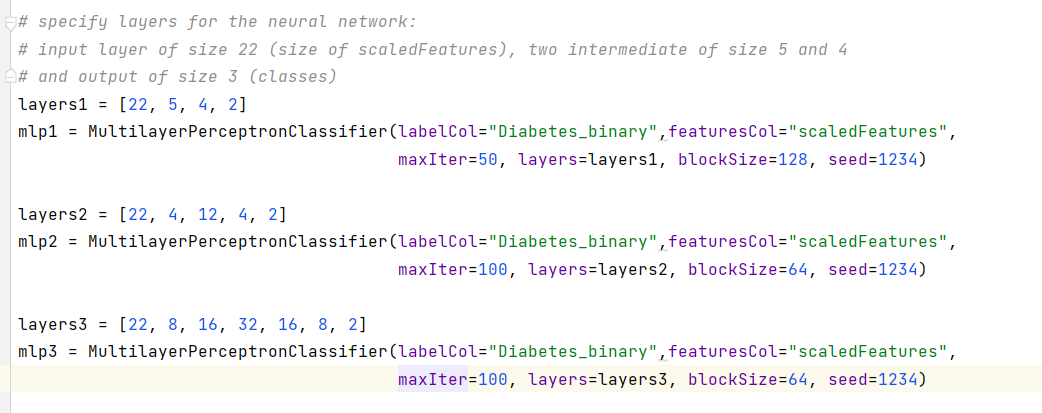
Првиот е LogisticRegression, каде ги менував бројот на итерации за тренирање и параметарот за регулација кој е константен. Се подесува името на колоната за класа, како и фамилија на проблемот – во овој случај бинарен излез.



Вториот модел е RandomForestClassifier, во кои сетирав број на дрва за одлучување, најголема длабочина на дрвото, како и мерка за мерење на „чистота„ на дрвото – gini or entropy. Поставена е документацијата за последниот параметар, кој се однесува на тоа колку колони ќе се земат при одлука.



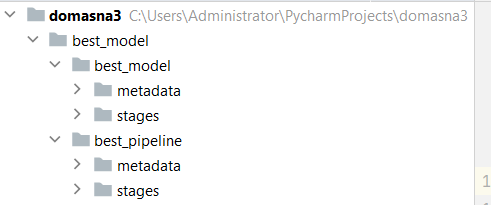
Последниот модел е MultiLayerPerceptron, односно невронски мрежи. Во низа се поставуваат големините на секој од слоевите и колку слоеви сакаме да има. ПРвиот број е секогаш ист бидејќи претставува влезни карактеристики, а последниот излезните класи. Освен слоевите, треба да се подеси и макс итерации како и големина на блок, што е обично степен од 2.



За секој од моделите се креира Pipeline, кој ги врши горните трансформации и го тренира соодветниот модел, а потоа го тестира истиот на test датасетот. Оној со најдобар Ф1 score се зачувува за да може да се употреби во следната фаза.



Зачуваниот модел и pipeline.



Конзола со Ф1:

F1 of LogisticRegression\_37fe58de9b9a is = 0.9999753618700554

F1 of LogisticRegression\_0701a12edc64 is = 0.8437595625268478

F1 of LogisticRegression\_91f417df490d is = 0.7962309277367923

F1 of RandomForestClassifier\_6613ef8ac64c is = 0.9767965308120141

F1 of RandomForestClassifier\_0fcd511b511e is = 1.0

F1 of RandomForestClassifier\_4441f843c7d8 is = 1.0

F1 of MultilayerPerceptronClassifier\_167da9a69843 is = 0.999975363695787

F1 of MultilayerPerceptronClassifier\_395511426f66 is = 1.0

F1 of MultilayerPerceptronClassifier\_ffb5c945fd27 is = 0.999975363695787

Барањe:

Online фаза.

Потребно е да направите producer python скипта (слично на ова скрипта) која ќе ги праќа податоците од датотеката offline.csv (ред по ред) во JSON формат на Apache Kafka topic health\_data. Да не се испраќа полето кое ја означува класата на пациентот! За подигнување на Apache Kafka и Zookeeper користете ги скриптите од првата домашна задача.

Во нова Apache Spark апликација, со помош на Spark Structured Streaming, потребно е да го вчитате потокот на податоци од topic-от health\_data. Врз вчитаните податоци потребно е да ги извршите истите трансформации како во offline фазата. Секој ред/запис од овој DataFrame потребно е да го предадете на моделот (којшто сте го обучиле во offline фазата и сте го вчитале на почетокот на ова апликација) и записите да ги збогатите (enrich) со предвидената класа од моделот. Вака добиените записи потребно е да ги испратите на нов topic health\_data\_predicted.

Решение:

Kafka Producer-от се креира да испраќа JSON податоци кон health\_data oд датотеката online.csv. Колоната за класата се отфрла, а праќањето е симулирано на рандом време.



За онлајн фазата се креира нова Спарк апликација. Се подесува форматот на stream-от, адреса и порта на која ќе слуша. Опцијата subscribe е подесување на topic-от.



Се вчитува најдобриот модел кој го зачувавме, како и Pipeline-от за да се направат трансофрмациите.



За секој batch од податоци во потокот, се повикува функцијата func\_call, која го претвора JSON во датафраме и го применува моделот за да креира предикција. Крајниот резултат се праќа на health\_data\_predicted.

